

Linux FHS

(Filesystem Hierarchy Standard)

LinuXpert Systems

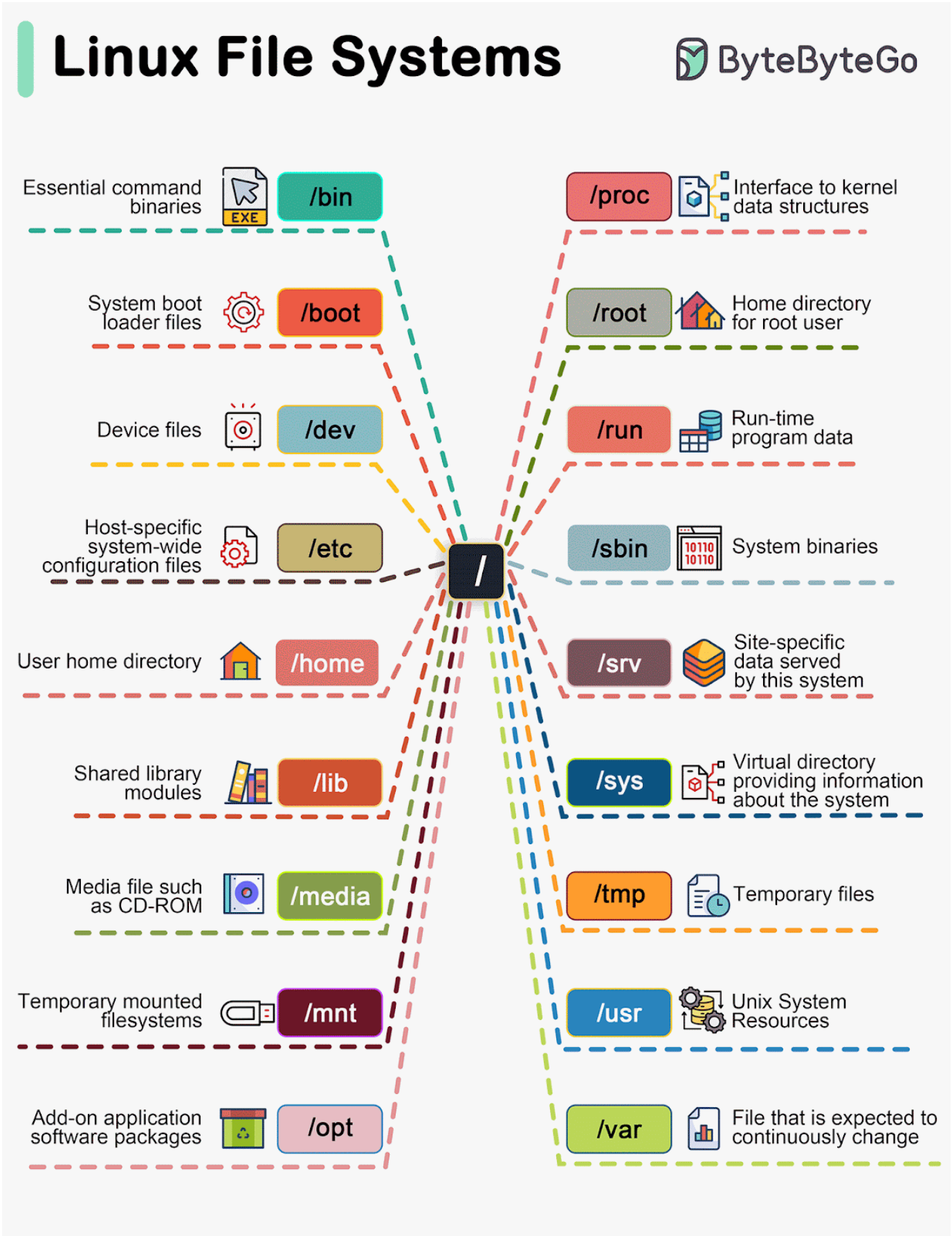
WhatsApp: 98841 65649, 90808 01382

E-mail: baskar@linuxpert.in

TABLE OF CONTENTS

1. What is the Linux filesystem hierarchy ? 2
2. The Filesystem Hierarchy Standard 3
3. The root directory / 4
4. /bin - Essential user commands 4
5. /sbin - System administration commands 5
6. /boot - Files required to boot Linux 6
7. /dev - Device files 7
8. /etc - System configuration 8
9. /home - Users' home directories 9
10. /root - Root user's home directory 10
11. /lib - Essential shared libraries and kernel modules ... 10
12. /media - Removable media 11
13. /mnt - Temporary mount point 12
14. /opt - Optional/add-on software 12
15. /proc - Process and kernel information 13
16. /run - Runtime system data 14
17. /srv - Data served by services 14
18. /sys - Kernel / device information 15
19. /tmp - Temporary files 16
20. /usr - User-space programs and data 16
21. /var - Variable data 19
22. /var/log - System logs 19
23. /var/cache - Cached data 20
24. /var/lib - Persistent application / system state 20
25. /var/spool - Queued jobs 21

26. /var/tmp - Persistent temporary files 21
27. The complete traditional FHS map 22
28. A much better way to memorize it 22
29. Physical vs Virtual filesystems 24
30. Understanding mounts 25
31. Finding where a command comes from 26
32. Finding shared libraries 26
33. Finding files efficiently 27
34. Linux Filesystem Hierarchy from an Administrator's perspective 28
35. Modern Linux: /usr merge 29
36. FHS vs Linux filesystem implementation 29
37. A practical laboratory exercise 30
38. The important commands to learn alongside FHS 32
39. A concise FHS cheat sheet 34


Linux File Systems
ByteByteGo
/
/bin
Essential command binaries
/boot
System boot loader files
/dev
Device files
/etc
Host-specific system-wide configuration files
/home
User home directory
/lib
Shared library modules
/media
Media file such as CD-ROM
/mnt
Temporary mounted filesystems
/opt
Add-on application software packages
/proc
Interface to kernel data structures
/root
Home directory for root user
/run
Run-time program data
/sbin
System binaries
/srv
Site-specific data served by this system
/sys
Virtual directory providing information about the system
/tmp
Temporary files
/usr
Unix System Resources
/var
File that is expected to continuously change

1. What is the Linux filesystem hierarchy ?

One of the most important concepts in Linux is that **everything starts from a single root directory, `/`**.

Unlike Windows, where you typically see drives such as:

```
C:\
D:\
E:\
```

Linux presents the system as one unified directory tree:

```
/
├── bin
├── boot
├── dev
├── etc
├── home
├── lib
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```

The filesystem hierarchy provides a **standardized organization of files and directories** so that users, administrators, applications, package managers and system utilities know where different types of files normally belong.

The historical standard for this organization is the **Filesystem Hierarchy Standard (FHS)**.

2. The Filesystem Hierarchy Standard

The **Filesystem Hierarchy Standard (FHS)** was developed to establish conventions for the directory structure of Unix-like operating systems.

It answers questions such as:

- Where should executable programs go?
- Where are system configuration files stored?
- Where are user files stored?
- Where should temporary files go?
- Where are logs stored?
- Where are kernel-related files exposed?
- Where should locally installed software reside?

The important point is:

FHS describes conventions, not an absolute rule that every Linux distribution must follow.

Modern distributions, particularly systems using `systemd`, have evolved beyond some of the original distinctions.

For example, on modern Ubuntu:

```
/bin  → /usr/bin
/sbin → /usr/sbin
/lib  → /usr/lib
```

may be symbolic links because of the **usr-merge** approach.

So when learning FHS, distinguish between:

FHS concept

and

what a particular modern Linux distribution actually implements.

3. The root directory `/`

The `/` directory is the **top of the entire Linux filesystem hierarchy**.

Everything accessible through the filesystem is ultimately located somewhere below `/`.

For example:

```
/home/baskar
/etc/ssh/sshd_config
/usr/bin/gcc
/var/log/syslog
```

All of these begin at `/`.

You can see the root directory with:

```
ls /
```

For a more useful view:

```
ls -lah /
```

---

4. `/bin` - Essential user commands

Historically:

```
/bin
```

contains **essential executable programs required by ordinary users and basic system operation**.

Examples traditionally include:

```
/bin/cp
/bin/mv
/bin/rm
/bin/cat
/bin/ls
/bin/sh
```

You can examine it with:

```
ls /bin
```

On many modern distributions:

```
ls -ld /bin
```

may show something like:

```
/bin -> usr/bin
```

This is because modern Linux distributions increasingly use **merged `/usr`**.

Important distinction

Do not interpret `/bin` as simply "all binary files."

The name means **binary executable programs**, in the historical Unix sense.

---

5. `/sbin` - System administration commands

Historically:

```
/sbin
```

contains programs primarily intended for **system administration**.

Examples include:

```
/sbin/fsck
/sbin/reboot
/sbin/shutdown
/sbin/mount
```

Modern systems may have:

```
/sbin -> usr/sbin
```

You can investigate:

```
ls -ld /sbin
```

and:

```
ls /sbin
```

6. `/boot` - Files required to boot Linux

The `/boot` directory contains files associated with the **Linux boot process**.

Typical contents:

```
/boot
├── vmlinuz-...
├── initrd.img-...
├── config-...
├── System.map-...
└── grub/
```

Important files

`vmlinuz`

Compressed Linux kernel image.

Example:

`/boot/vmlinuz-6.8.0-xx-generic`

`initrd` / `initramfs`

Initial RAM filesystem used during early boot.

Example:

`/boot/initrd.img-6.8.0-xx-generic`

`grub`

Bootloader configuration and supporting files.

For example:

`/boot/grub/`

Useful commands

```
ls -lh /boot
```

Check the running kernel:

```
uname -r
```

List installed kernels:

```
ls /boot/vmlinuz*
```

---

7. `/dev` - Device files

This is one of the most interesting parts of Linux.

```
/dev
```

contains **device nodes** representing hardware and virtual devices.

Examples:

```
/dev/sda
/dev/nvme0n1
/dev/tty
/dev/null
/dev/zero
/dev/random
/dev/urandom
```

Linux traditionally follows the philosophy:

Everything is a file.

Hardware devices can therefore be accessed through filesystem interfaces.

For example:

```
ls -l /dev/null
```

You may see:

```
crw-rw-rw- ... /dev/null
```

The **c** indicates a **character device**.

A disk might appear as:

```
brw-rw---- ... /dev/sda
```

where **b** means **block device**.

Important

`/dev` is generally a **virtual/dynamic filesystem**, commonly managed by `udev`.

It isn't simply a directory containing ordinary files permanently stored on your disk.

---

8. `/etc` - System configuration

If you remember only one directory for system administration, remember:

```
/etc
```

It contains **system-wide configuration files**.

Examples:

```
/etc/passwd
/etc/group
/etc/hostname
/etc/hosts
/etc/fstab
/etc/resolv.conf
/etc/ssh/
/etc/systemd/
/etc/apt/
```

`/etc/passwd`

Contains user account information.

```
cat /etc/passwd
```

`/etc/fstab`

Defines filesystems that should be mounted.

```
cat /etc/fstab
```

`/etc/hostname`

Contains the system hostname.

```
cat /etc/hostname
```

`/etc/hosts`

Local hostname-to-IP mappings.

```
cat /etc/hosts
```

`/etc/ssh`

SSH configuration.

```
ls /etc/ssh
```

---

9. `/home` - Users' home directories

Normally, ordinary users have their personal directories under:

```
/home
```

For example:

```
/home/baskar
/home/student
/home/admin
```

A user's home directory contains their personal:

```
Documents
Downloads
Pictures
Videos
configuration files
application data
```

For example:

```
/home/baskar/.bashrc
/home/baskar/.ssh/
```

The `.` prefix indicates hidden files/directories.

To see them:

```
ls -la ~
```

`~` means the current user's home directory.

---

10. `/root` - Root user's home directory

Do not confuse:

```
/
```

with:

```
/root
```

`/` is the filesystem root.

`/root` is the **home directory of the root user**.

For example:

```
/root/.bashrc
/root/.ssh/
```

The root user's home is separated from `/home` because root must be able to operate even when `/home` is unavailable or mounted separately.

---

11. `/lib` - Essential shared libraries and kernel modules

Historically:

```
/lib
```

contains essential shared libraries required by programs in `/bin` and `/sbin`.

It can also contain:

```
/lib/modules/
```

which contains Linux kernel modules.

For example:

```
ls /lib/modules
```

You may see:

```
6.8.0-xx-generic
```

Inside:

```
/lib/modules/<kernel-version>/
```

you can find kernel modules such as:

```
*.ko
*.ko.xz
*.ko.zst
```

Related commands

```
lsmod
modinfo
modprobe
insmod
rmmod
depmod
```

These are the **Linux kernel module management utilities** we discussed earlier.

---

12. `/media` - Removable media

The traditional purpose of:

```
/media
```

is to provide mount points for removable media.

Examples:

```
/media/usb
/media/baskar/USB_DISK
```

Modern desktop environments often automatically mount USB drives somewhere under `/media`.

You can see mounted filesystems with:

```
findmnt
```

or:

```
lsblk
```

---

13. `/mnt` - Temporary mount point

```
/mnt
```

is traditionally used for **temporarily mounted filesystems**.

For example:

```
sudo mount /dev/sdb1 /mnt
```

Now the filesystem becomes accessible through:

```
/mnt
```

Administrators may also create directories such as:

```
/mnt/data
/mnt/backup
/mnt/test
```

for temporary or administrative mounts.

---

14. `/opt` - Optional/add-on software

```
/opt
```

is traditionally intended for **add-on application software packages**.

For example:

```
/opt/myapplication
/opt/oracle
/opt/google
```

It is particularly useful for applications that don't fit neatly into the distribution's normal package layout.

15. `/proc` - Process and kernel information

`/proc` is extremely important for Linux administration.

It is a **virtual filesystem** that exposes information about:

- running processes
- kernel state
- CPU
- memory
- devices
- kernel parameters

For example:

```
ls /proc
```

You will see numeric directories:

```
1
2
3
...
```

These correspond to **process IDs (PIDs)**.

For example:

```
/proc/1
```

is information about PID 1.

`/proc/cpuinfo`

```
cat /proc/cpuinfo
```

Displays CPU information.

`/proc/meminfo`

```
cat /proc/meminfo
```

Displays memory information.

`/proc/uptime`

```
cat /proc/uptime
```

Displays system uptime information.

`/proc/mounts`

Shows mounted filesystems.

```
cat /proc/mounts
```

---

16. `/run` - Runtime system data

```
/run
```

contains **volatile runtime data** created since the system booted.

Examples include:

```
/run/systemd/
/run/lock/
/run/user/
/run/sshd/
```

This directory normally resides on a temporary filesystem such as `tmpfs`.

Therefore:

Data in `/run` normally disappears after reboot.

It is used extensively by `systemd` and other system services.

---

17. `/srv` - Data served by services

```
/srv
```

is intended for **data served by system services**.

For example, a web server could theoretically use:

```
/srv/www/
```

An FTP server might use:

```
/srv/ftp/
```

In practice, Linux distributions and applications frequently use other locations such as:

```
/var/www
```

instead.

So `/srv` is an FHS convention rather than something you will necessarily see heavily used.

---

18. `/sys` - Kernel/device information

`/sys` is another extremely important virtual filesystem.

It is generally associated with **sysfs**, which exposes information about:

- devices
- drivers
- buses
- kernel subsystems
- hardware topology

For example:

```
ls /sys
```

You'll find directories such as:

```
/sys/class
/sys/devices
/sys/block
/sys/bus
/sys/firmware
```

Example

```
ls /sys/class/net
```

might show:

```
eth0
lo
```

or:

```
enp3s0
lo
```

This is useful for understanding Linux hardware/device management.

19. `/tmp` - Temporary files

`/tmp`

is intended for temporary files.

Applications may create temporary data such as:

```
/tmp/example.tmp
/tmp/session-data
```

Many systems clean `/tmp` automatically.

Important

Do **not** assume that files in `/tmp` will survive a reboot.

If something is important, don't store it there.

---

20. `/usr` - User-space programs and data

This is one of the most misunderstood directories.

Historically, `/usr` was associated with **Unix system resources**, and despite its name it does **not simply mean "user files."**

A modern `/usr` commonly contains:

```
/usr
├── bin
├── sbin
├── lib
├── include
├── share
├── local
└── src
```

/usr/bin

Contains a very large collection of user-space executable programs.

Examples:

```
/usr/bin/gcc
/usr/bin/python3
/usr/bin/ssh
/usr/bin/grep
/usr/bin/find
```

Check:

```
which gcc
```

or:

```
command -v gcc
```

/usr/sbin

Contains system administration programs.

/usr/lib

Contains libraries and other architecture-dependent data.

/usr/include

Contains header files used for software development.

For example:

```
/usr/include/stdio.h
```

This is particularly relevant when compiling C/C++ programs.

/usr/share

Contains architecture-independent shared data.

Examples:

```
/usr/share/man
/usr/share/doc
/usr/share/icons
/usr/share/applications
/usr/share/locale
```

/usr/local

This deserves special attention.

It is intended for **software installed locally by the system administrator**, rather than software managed by the distribution's normal package system.

Typical directories:

```
/usr/local/bin
/usr/local/sbin
/usr/local/lib
/usr/local/share
```

For example, software manually installed by an administrator may place an executable in:

```
/usr/local/bin/myprogram
```

This is also why you often see manually installed tools under:

```
/usr/local/bin
```

21. `/var` - Variable data

The name comes from **variable**.

`/var` contains data that changes during normal system operation.

Typical structure:

```
/var
├── cache
├── lib
├── log
├── mail
├── spool
└── tmp
```

---

22. `/var/log` - System logs

One of the most important administration directories:

```
/var/log
```

Examples may include:

```
/var/log/syslog
/var/log/auth.log
/var/log/kern.log
```

On systemd systems, much logging is handled by `journald`.

You can inspect it with:

```
journalctl
```

Kernel messages:

```
journalctl -k
```

Recent errors:

```
journalctl -p err
```

23. `/var/cache` - Cached data

Applications and package managers may store cached data here.

For example, Debian/Ubuntu systems use:

```
/var/cache/apt/
```

You can examine:

```
du -sh /var/cache/*
```

---

24. `/var/lib` - Persistent application/system state

This is an extremely important directory.

```
/var/lib
```

contains **persistent state maintained by applications and system services**.

Examples:

```
/var/lib/dpkg
/var/lib/systemd
/var/lib/mysql
/var/lib/postgresql
```

For example, Debian's package database resides under:

```
/var/lib/dpkg
```

A database server may store its data under:

```
/var/lib/mysql
```

or another distribution/application-specific location.

25. `/var/spool` - Queued jobs

`/var/spool` contains data waiting to be processed.

Examples include:

```
/var/spool/cron
/var/spool/mail
```

The concept is:

```
generate → queue → process
```

For example, a print job can be placed into a spool until the printer processes it.

---

26. `/var/tmp` - Persistent temporary files

Both:

```
/tmp
```

and:

```
/var/tmp
```

are temporary storage locations.

But traditionally:

```
/tmp
```

is more transient, while:

```
/var/tmp
```

is intended for temporary files that may need to survive across reboots.

This distinction is useful when writing system administration scripts.

27. The complete traditional FHS map

A useful way to remember the hierarchy is:

```
/
├── bin      Essential user commands
├── boot     Bootloader + kernel files
├── dev      Device files
├── etc      System configuration
├── home     Users' home directories
├── lib      Essential libraries + kernel modules
├── media    Removable media mount points
├── mnt      Temporary mount points
├── opt      Optional/add-on software
├── proc     Process/kernel virtual filesystem
├── root     Root user's home
├── run      Runtime system data
├── sbin     System administration commands
├── srv      Service data
├── sys      Kernel/device virtual filesystem
├── tmp      Temporary files
├── usr      User-space programs/data
└── var      Variable/persistent changing data
```

---

28. A much better way to memorize it

Rather than memorizing 20 directory names individually, group them by **purpose**.

Boot

`/boot`

Think: "How does Linux start?"

Configuration

`/etc`

Think: "How is Linux configured?"

Users

```
/home
/root
```

Think: "Who owns personal files?"

Programs

```
/bin
/sbin
/usr/bin
/usr/sbin
/usr/local/bin
```

Think: "Where are executable programs?"

Libraries

```
/lib
/usr/lib
/usr/local/lib
```

Think: "Where does executable code get its shared libraries?"

Hardware/kernel interfaces

```
/dev
/proc
/sys
```

Think:

```
/dev  → devices
/proc → processes/kernel information
/sys  → devices/kernel objects
```

Temporary/runtime

```
/tmp
/var/tmp
/run
```

Changing system data

```
/var
```

Mounting external filesystems

```
/media
/mnt
```

---

29. Physical vs virtual filesystems

This distinction is extremely important.

Not everything you see under `/` is an ordinary directory stored on your disk.

For example:

Directory	Nature
`/etc`	Real filesystem data
`/home`	Real filesystem data
`/usr`	Real filesystem data
`/var`	Real filesystem data
`/dev`	Virtual/dynamic
`/proc`	Virtual
`/sys`	Virtual
`/run`	Runtime/volatile

You can investigate this with:

```
findmnt
```

or:

```
df -Th
```

For example:

```
df -Th / /proc /sys /dev /run
```

This gives you a practical demonstration of how different parts of the hierarchy are implemented.

---

30. Understanding mounts

One of Linux's most powerful filesystem concepts is that **a filesystem can be attached anywhere in the directory tree**.

For example:

```
/
├── home
├── var
├── usr
└── data
```

You could mount another disk:

```
sudo mount /dev/sdb1 /data
```

and suddenly:

```
/data
```

represents the contents of `/dev/sdb1`.

This is why Linux doesn't need drive letters.

You can have:

```
/
/home
/var
/data
/backup
```

all backed by different filesystems.

Check them with:

```
findmnt
```

or:

```
lsblk -f
```

31. Finding where a command comes from

The filesystem hierarchy becomes much more meaningful when you investigate real commands.

For example:

```
which ls
```

or:

```
command -v ls
```

You might get:

```
/usr/bin/ls
```

Then:

```
ls -l /usr/bin/ls
```

You can also use:

```
type ls
```

---

32. Finding shared libraries

For an executable:

```
ldd /usr/bin/ls
```

This connects directly with the GNU binary inspection tools you have been studying.

You can see which shared libraries the executable depends on.

For example:

```
libc.so.6
libselinux.so.1
libpcre2-8.so.0
...
```

Those libraries are generally located somewhere under:

```
/lib
/usr/lib
```

This gives you a practical connection:

```
/usr/bin/ls
      │
      └── shared libraries
                │
                ├── /lib
                └── /usr/lib
```

---

33. Finding files efficiently

Once you understand the hierarchy, these commands become essential.

`find`

```
find /etc -type f
```

Find files under `/var`:

```
find /var -type f
```

Find files larger than 500 MB:

```
find / -type f -size +500M 2>/dev/null
```

`du`

Directory usage:

```
du -sh /var
```

Largest directories:

```
du -h --max-depth=1 /var | sort -h
```

`df`

Filesystem capacity:

```
df -h
```

`lsblk`

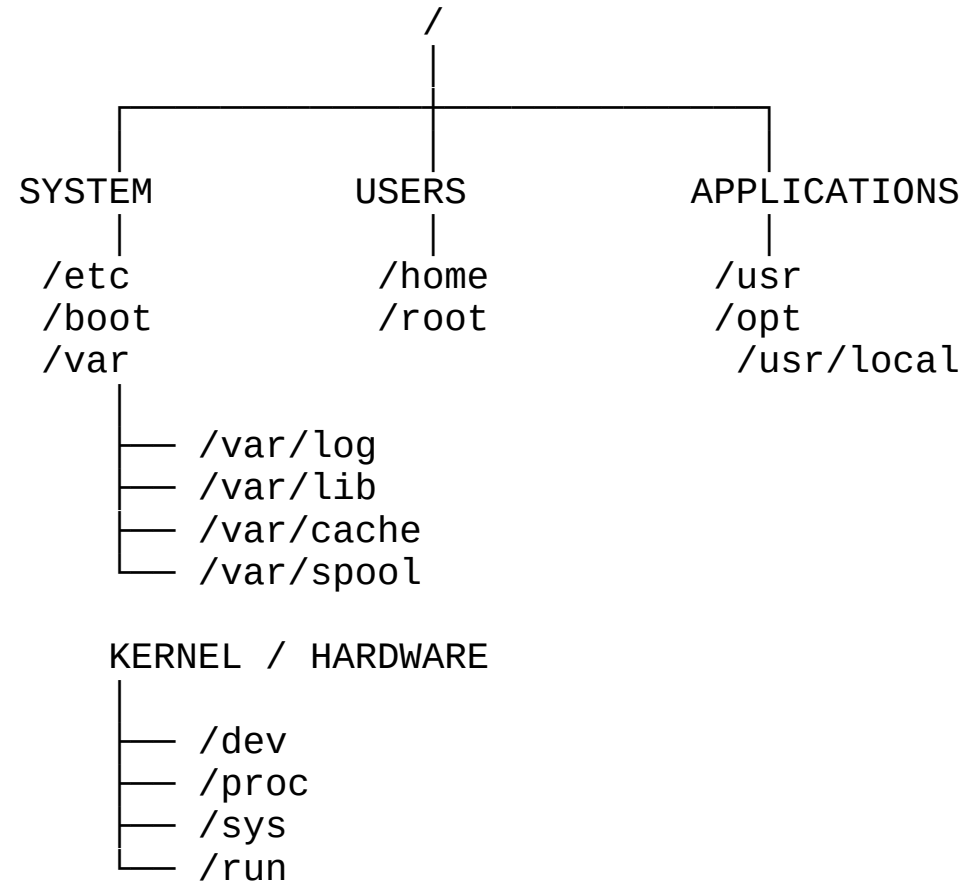
Block devices:

```
lsblk -f
```

---

34. Linux filesystem hierarchy from an administrator's perspective

A useful mental model is:



This is much easier to remember than memorizing directory names individually.

35. Modern Linux: `/usr` merge

If you are teaching this today, **do not omit this topic**.

On many current distributions, including Ubuntu, the traditional directories:

```
/bin
/sbin
/lib
```

have effectively been consolidated into `/usr`.

For example:

```
/bin  → /usr/bin
/sbin → /usr/sbin
/lib  → /usr/lib
```

You can check your system:

```
ls -ld /bin /sbin /lib
```

If they are symbolic links, you'll see something like:

```
/bin -> usr/bin
```

This is known as **usr-merge**.

So the traditional FHS diagram is still valuable for understanding Linux, but a modern Linux administrator should know that the physical implementation may differ.

---

36. FHS vs Linux filesystem implementation

Another important distinction:

FHS is not the Linux filesystem itself.

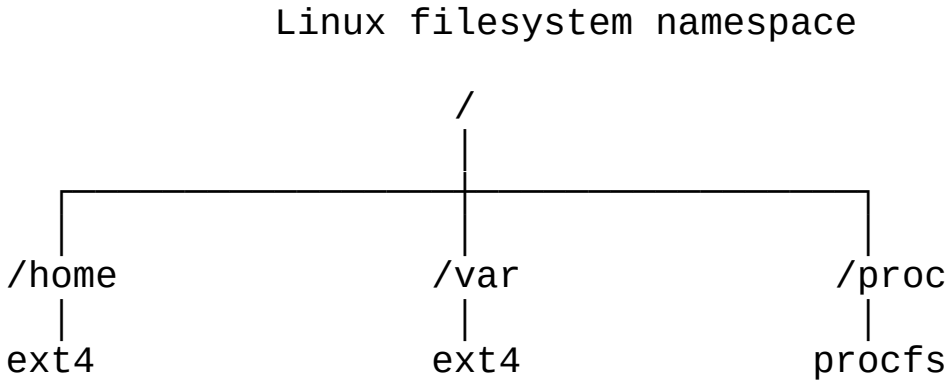
Linux supports many filesystem types:

```
ext4
XFS
Btrfs
ZFS
NFS
tmpfs
proc
sysfs
overlayfs
...
```

FHS describes **where things are organized in the namespace**.

The underlying filesystem determines **how the data is actually stored**.

For example:



So `/proc` is not an ordinary ext4 directory containing process files. It is a filesystem interface generated by the kernel.

---

37. A practical laboratory exercise

If you're teaching Linux, I strongly recommend giving students this exercise.

Run:

```
cd /
```

Then:

```
pwd
```

You should get:

```
/
```

Now:

```
ls
```

Then investigate each major directory:

```
ls -ld /bin /boot /dev /etc /home /lib /media /mnt /opt /proc
/root /run /sbin /srv /sys /tmp /usr /var
```

Next:

```
df -Th
```

Then:

```
findmnt
```

Then:

```
lsblk -f
```

Then investigate the kernel:

```
uname -a

ls /proc
ls /sys
ls /dev
```

Finally, investigate an executable:

```
which ls

file $(which ls)

ldd $(which ls)
```

This single exercise connects:

```
FHS
  │
  ├── filesystem
  ├── kernel
  ├── devices
  ├── processes
  ├── executables
  ├── shared libraries
  └── system administration
```

38. The most important commands to learn alongside FHS

For a Linux learner, organize and learn the commands like this:

Navigation

```
pwd
ls
cd
tree
```

File management

```
cp
mv
rm
mkdir
touch
ln
```

Searching

```
find
locate
which
whereis
```

Filesystem / Storage

```
df
du
mount
umount
findmnt
lsblk
blkid
```

Permissions

```
chmod
chown
chgrp
umask
```

Filesystem information

```
stat
file
```

Kernel / Filesystem interfaces

```
lsmod
modprobe
dmesg
sysctl
ls /proc
ls /sys
ls /dev
```

39. A concise FHS cheat sheet

Path	Remember it as
`/`	Root of everything
`/boot`	Boot files
`/dev`	Devices
`/etc`	Configuration
`/home`	Users
`/root`	Root user's home
`/lib`	Essential libraries/modules
`/media`	Removable media
`/mnt`	Temporary mounts
`/opt`	Optional software
`/proc`	Processes / kernel interface
`/run`	Runtime data
`/srv`	Service data
`/sys`	Kernel / device interface
`/tmp`	Temporary files
`/usr`	User-space programs/data
`/var`	Variable / changing data

The easiest mnemonic

```
/boot   →   Boot
/dev    →   Devices
/etc    →   Configuration
/home   →   Users
/lib    →   Libraries
/opt    →   Optional software
/proc   →   Processes
/root   →   Root user
/run    →   Runtime
/sbin   →   System administration
/srv    →   Services
/sys    →   System/kernel
/tmp    →   Temporary
/usr    →   Programs/resources
/var    →   Variable data
```